Capstone Project: Using a Machine Learning Model to Classify Distal Radius Fractures

A. Laurie Wells, PhD

College of Science, Engineering and Technology, Grand Canyon University

DSC 590: Data Science Capstone Project

Chintan Thakkar

March 2, 2022

Capstone Project

General Information

Project name: Using a Machine Learning Model to Classify Distal Radius Fractures

Author: Ann Laurie Wells

Project organization: Department of Orthopaedics and Rehabilitation, University of New Mexico, School of Medicine

Project manager: Deana Mercer, MD, Principal Investigator

Date project proposal form is submitted: Initial: August 18, 2021, Final: January 19, 2022

Project Overview and Objectives

State the problem

Orthopedic surgeons at the University of New Mexico (UNM) School of Medicine (SOM) have developed a classification system for distal radius fractures intended to aid clinicians in choosing appropriate treatment.

Three fellowship trained hand surgeons classified the initial 300 cases. After the classification, 13 of the cases were removed and replaced. In some of the cases the break was judged to be a shaft fracture, not a distal radius fracture. One case was determined to have pre-existing DRUJ damage. Table 1 shows the intra-rater and inter-rater reliability for the three experienced surgeons for the 287 cases they evaluated. For the classification system to be useful, consistent and accurate classification is necessary. If a machine learning model with sufficient accuracy can be established, it would be a useful diagnostic aid in providing consistent classifications.

Table 1. Intra-rater and inter-rater reliability

Classifier	Fleiss' Kappa for Categorical Data	Agreement
Surgeon 1	0.798	Good
Surgeon 2	0.665	Moderate
Surgeon 3	0.617	Moderate
Inter-rater reliability	0.564	Moderate

Background

Distal radius fractures are the most common fractures in adults (Court-Brown & Caesar, 2006). Frequently, these fractures are treated surgically (Rundgren et al., 2020). The stability of the distal radio-ulnar joint (DRUJ) is critical to achieving good clinical outcomes for patients (Geissler, Fernandez, & Lamey, 1996). Factors that affect stability of the DRUJ after a distal radius fracture may include a concomitant ulnar fracture, complete triangular fibrocartilage complex (TFCC) tears, and loss of tension in the distal interosseous ligament (DIOL) (Mercer et al., 2021).

This project is part of a larger research effort to study these injuries. Current thinking is that there is a high incidence of DRUJ instability in distal radius fractures after anatomical restoration and stable fixation. The hypothesis of the larger study is that this is only true for certain types of fractures. The overarching study plan is to show that proper classification of fractures could lead to better treatment decisions.

Project Objectives

The objective of this project is to train, validate, and test a machine learning model to classify fractures from plain x-rays using the new classification system.

The three classes are defined by Mercer et al. (2021) as follows.

Type I: The triangular fibrocartilage complex (TFCC) and the distal portion of the interosseous forearm ligament or distal interosseous ligament (DIOL) remain intact after the

distal radius fracture (DRF). There is no residual instability or subluxation of the distal radio-ulnar joint (DRUJ) after anatomical reduction of the skeletal structures. This is the injury found in minimally displaced DRF's and in fractures of both radius and ulna which occur just proximal to the DIOL. They need no specific treatment besides restoration of the bony anatomy.

Type II: The TFCC and the extensor carpi ulnaris (ECU) tendon sheath rupture but the DIOL remains intact. DRUJ subluxation is corrected and adequate stability is restored after anatomical reduction of the skeletal structures. The distal ulna and/or the ulnar styloid may or may not be fractured. This is the concomitant DRUJ injury found in most displaced DRF's.

Type III: The TFCC, the ECU tendon sheath and the DIOL all rupture. Therefore, all ligamentous support for the DRUJ is lost. After anatomical reduction of the skeletal structures, either subluxation of the DRUJ persists or clinical testing shows DRUJ instability. It is necessary to address the persisting DRUJ instability by specific means and early forearm rotation is usually not possible. This is the concomitant DRUJ injury found in the Galeazzi fracture, fractures of the distal radius with radio-ulnar diastasis and some high energy comminuted distal radius and ulna fractures.

Figure 1, retrieved from https://osteopathy.colganosteo.com/triangular-fibrocartilage-complex-tfcc-injuries/, shows the TFCC and the ECU tendon sheath. Figure 2, retrieved from https://www.cureus.com/articles/36962-galeazzi-fracture-dislocations-an-illustrated-review, shows the DIOL.

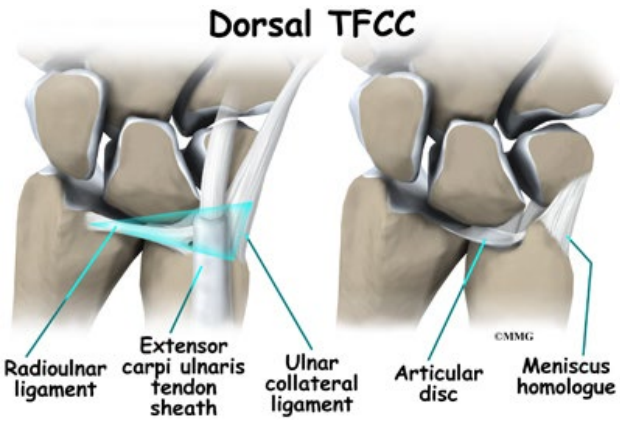


Figure 1. TFCC and ECU tendon sheath

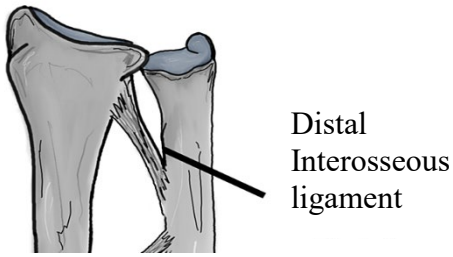


Figure 2. Distal Interosseous ligament (DIOL)

Challenges

The primary challenge for this project is the low number of classified x-rays that are available for training the model. Because this is a new classification system, the only classified x-rays will be the ones done for the study. An additional challenge is that the data is not balanced. Of the 300 classified cases, 73 are type I, 215 are type II, and 12 are type III. Figure 3 is a bar chart of the types of fracture. During training of the model, the training and validation was stratified by classification.

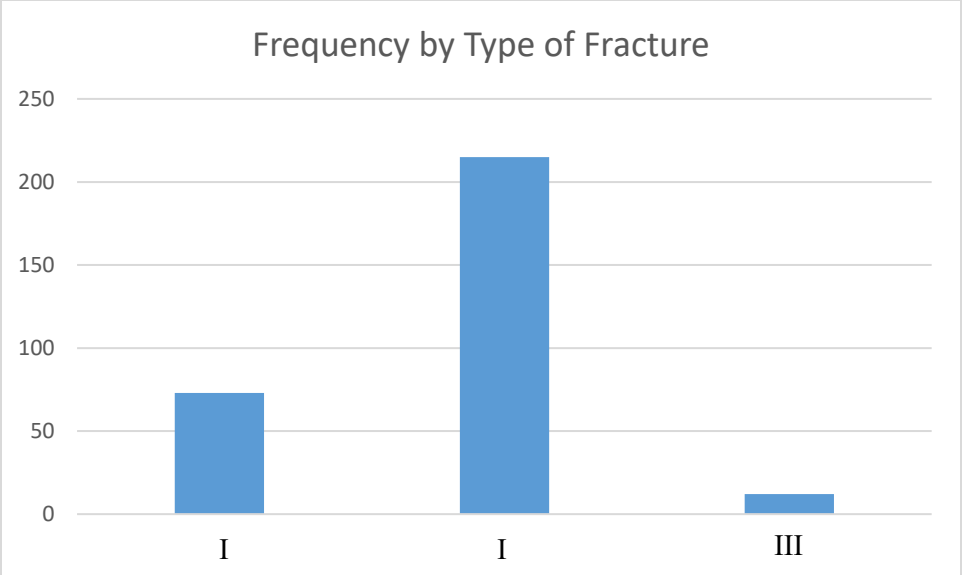


Figure 3. Frequency by type of fracture in training/validation set.

Benefits and Opportunities

Machine learning in medical imaging diagnostics is a growing field that is taking advantage of the rapidly expanding available data (El-Baz, Gimel'farb, & Suzuki, 2017). A machine learning tool for classifying distal radius fractures will benefit patients in situations where a highly experienced, fellowship-trained upper extremity orthopedic surgeon is not available to diagnose the distal radius fracture.

Project Scope

The scope of this project is limited to a machine learning task. We have 300 classified, high-resolution x-rays with which to work. We used a convolutional neural network for the classification.

Project Schedule

Figure 4 is a Gantt chart of the schedule. The Gantt chart was completed using the student version of TeamGantt.

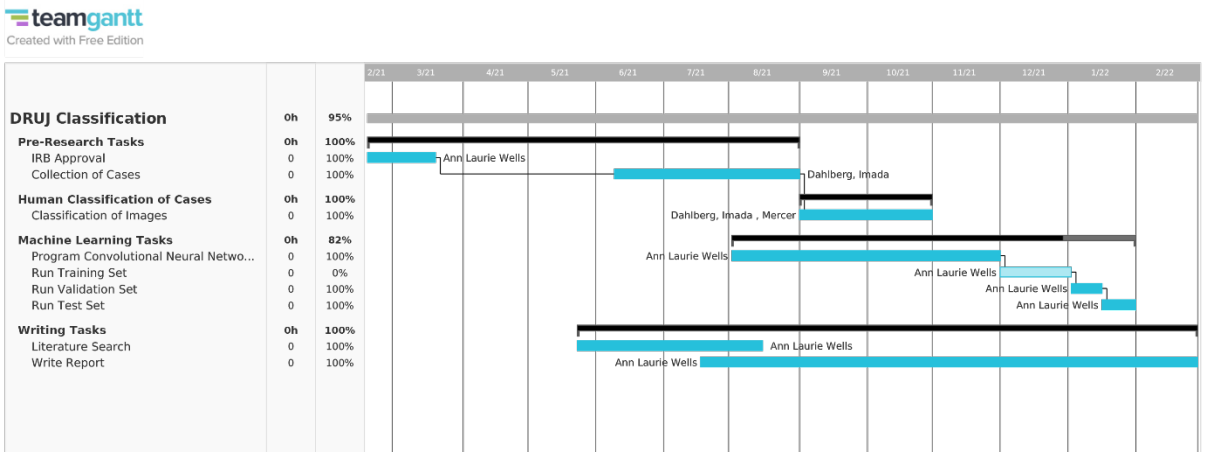


Figure 4. Gantt chart of the project schedule.

Cost Estimate

The cost of this project was covered by the Department of Orthopaedics and Rehabilitation research budget. It was essentially the time of the researchers. The records and computer resources required were already on hand.

Requirements Analysis

Use Cases

The expectation is that the model generated in this project will be used by doctors who have a set of digital x-rays of a distal radius fracture. A doctor will use the x-rays as the input. The output will be a classification: I, II, or III.

System Design

Once the model was trained, validated, and tested, the parameters and weights of the model were saved. At this point the user needs to have Python installed. There is a Python script to load the parameters and weights of the model and the x-ray. The output of the script is a classification. The follow-on plan to this project is to 1) get more training data and 2) build a compiled executable that will not require the user to run a Python script.

Technical Requirements

The convolutional neural network model was built using Python with the Keras and TensorFlow packages. The model was run on a Dell Precision 7750 workstation.

Data Science Model

A diagram of a convolutional network for medical image analysis from Kandell & Castelli (2020) is shown in figure 5.

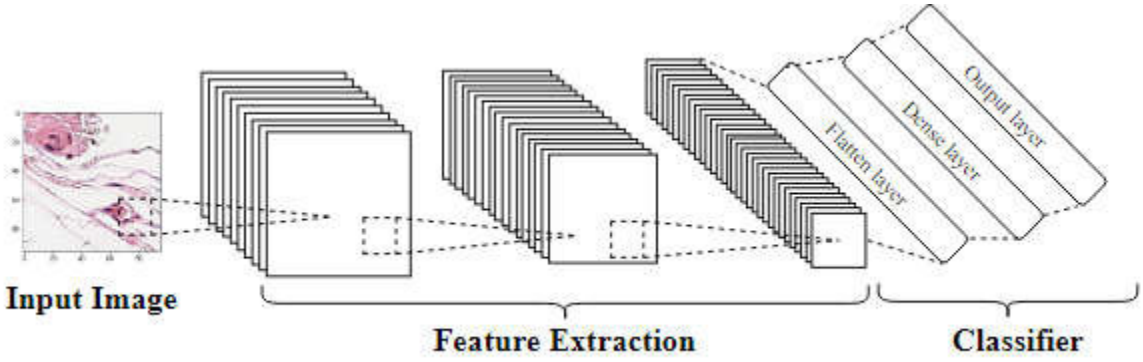


Figure 5. Convolutional neural network diagram.

Design Planning Summary

In a previous study done in the Department of Orthopaedics and Rehabilitation at the UNM SOM classifying fifth metatarsal fractures, both inter- and intra-rater reliability measures were low, particularly for less experienced surgeons (Packard et al., 2020). In the current study, we had three fellowship-trained upper extremity surgeons with over ten years of experience each classify the fractures. We expected their classification of the x-rays to be highly accurate. We also had two orthopedic residents with much less experience. We expected them to have more trouble classifying the x-rays. For the classification system to be useful, consistent and accurate classification is necessary. If a machine learning model with sufficient accuracy can be established, it would be a useful diagnostic aid in situations where a highly experienced, fellowship-trained surgeon is not available.

Benefits and Opportunities

Machine learning in medical imaging diagnostics is a growing field that is taking advantage of the rapidly expanding available data (El-Baz, Gimel'farb, & Suzuki, 2017). A machine learning tool for classifying distal radius fractures will benefit patients in situations where a highly experienced, fellowship-trained upper extremity orthopedic surgeon is not available to diagnose the distal radius fracture.

Project Scope

The scope of this project is limited to a machine learning task. We had 300 classified cases. Each case had three high-resolution x-ray views of the fracture with which to work. We used a convolutional neural network for the classification.

Model Pipeline Design

Data

The first step was to obtain permission from the University of New Mexico Health Sciences Center (UNM HSC) Internal Review Board (IRB) to conduct human research. The IRB granted approval March 17, 2021.

The next step was to search the UNM HSC electronic medical record system for suitable cases. The period we were given to search was a ten-year period, January 1, 2011 – December 31, 2020. The search was done based on Current Procedural Terminology (CPT) codes. The search came back with 1322 potential subjects.

The IRB granted approval to download up to 400 records. We currently have 300 records for which there were three x-ray views of the wrist from the original injury, posterioanterior (PA) view, oblique view, and lateral view. Figure 6 shows the three views for a single case. For some candidates, the first x-rays in the UNM system were taken after the fracture was reduced and casted. These do not meet the requirements for this study. Other candidates were eliminated because they were pediatric patients (< 18 years-old at the time of injury). We were not allowed to download images or information for those patients.

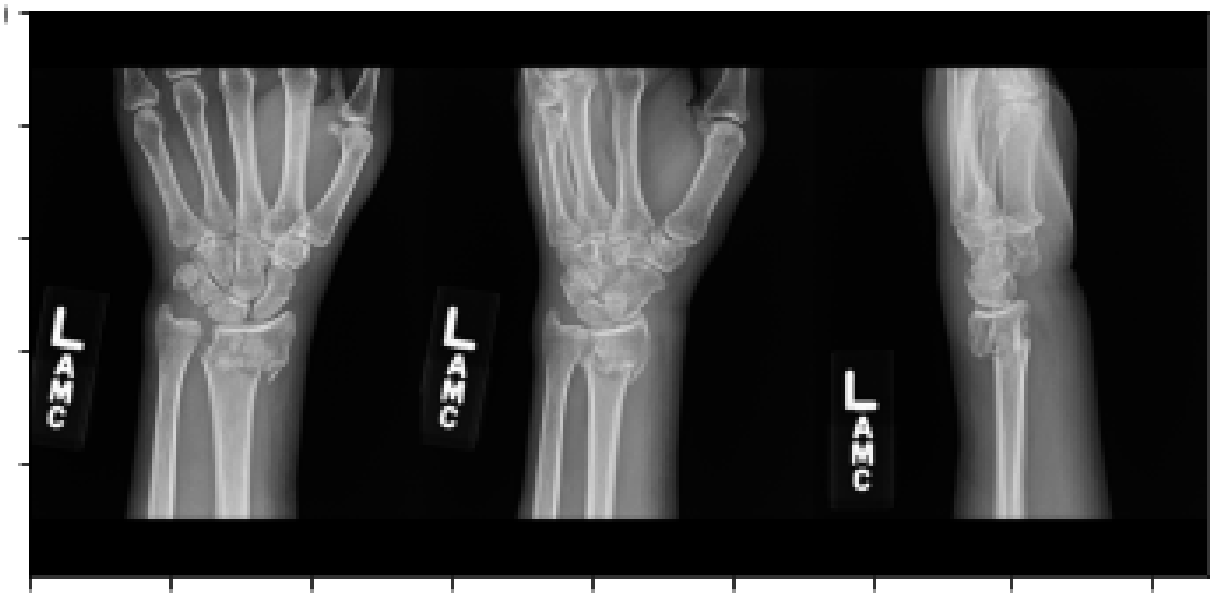


Figure 6. Three views of a single case, PA, oblique, and lateral.

After candidates were found that met the inclusion criteria, de-identified DICOM images were downloaded from PACS. Even though the images were de-identified, they were stored securely based on Health Insurance Portability and Accountability Act (HIPAA) requirements. The images were stored on a UNM HSC owned Dell Precision 7750 portable machine learning workstation and backed up to UNM HSC cloud storage. The images were processed in a Python Jupyter notebook using the *pydicom* package. The image pixels were converted to grayscale values, 0 to 255, inclusive. The images were cropped to 1000 x 700 pixels and saved to a csv file. Figure 7 shows an uncropped image and the corresponding cropped image. Before modelling, the data was normalized by converting the values to floating point and dividing by 255 to make numbers from 0 to 1 (Yadav & Jadhav, 2019).

The training/validation set was 270 classified cases. It was a rank-4 tensor with dimensions 270 x 1000 x 700 x 3. The test set was made up of 30 cases (30 x 1000 x 700 x 3). For running the model, the three images for each case were concatenated so the dimensions of the file used for the model was 270 x 1000 x 2100 x 1.

Images for the 300 cases were saved in a PowerPoint presentation with one slide for each case. The surgeons used the PowerPoint presentation to classify the fractures.

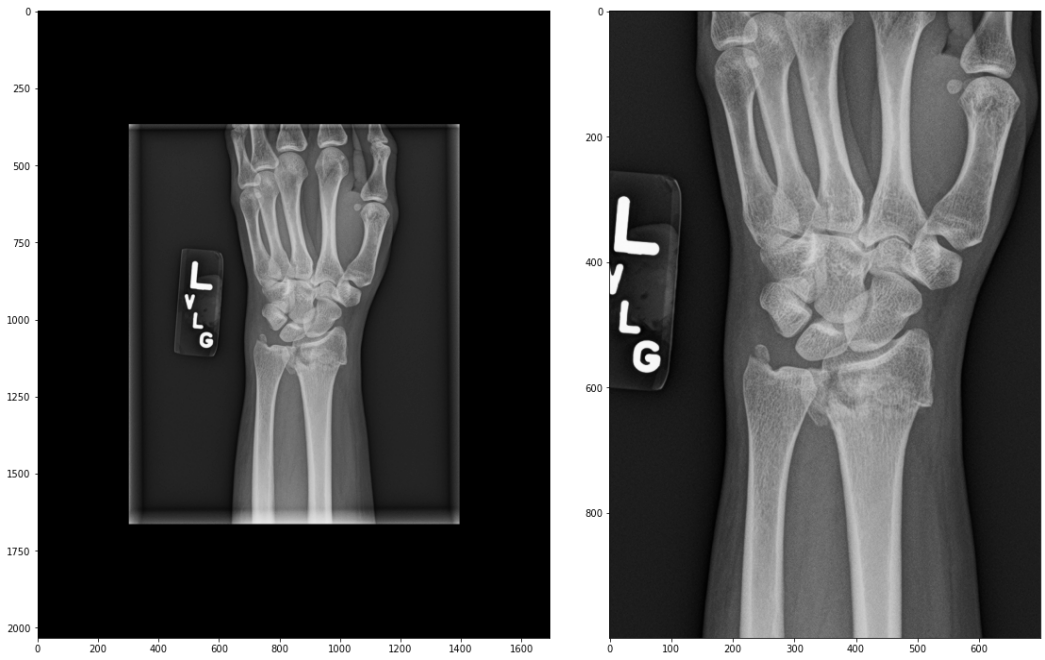


Figure 7. Uncropped image 2050 x1700 pixels with cropped image, 1000 x 700 pixels.

Software

A convolutional neural network (CNN) was used for the model. CNNs are a standard technology for classifying medical images (Yadav & Jadhav, 2019). The model was run using Python 3. It was built in a Jupyter notebook using the *keras* package.

Once I began running the CNN, I found that I did not have enough memory available, and the Python kernel crashed. I reduced the size of the images by ¼ by averaging 2 x 2 groups of pixels. This gave me dimensions for the training/validation set of 270 x 500 x 1050 x 1 and for the test set dimensions of 30 x 500 x 1050 x 1. Figure 8 compares the original full resolution image with the reduced resolution image.

Hardware

The model was processed on the previously mentioned Dell Precision 7750. The computer has 64 gigabytes of memory, 12 Intel(R) Core (TM) i7-10750 H CPU, 2.60 GHz processors, an NVIDIA Quadro RTX 3000 GPU, and a 1 terabyte solid state hard drive. There is also an external 5 terabyte solid state hard drive available if needed.

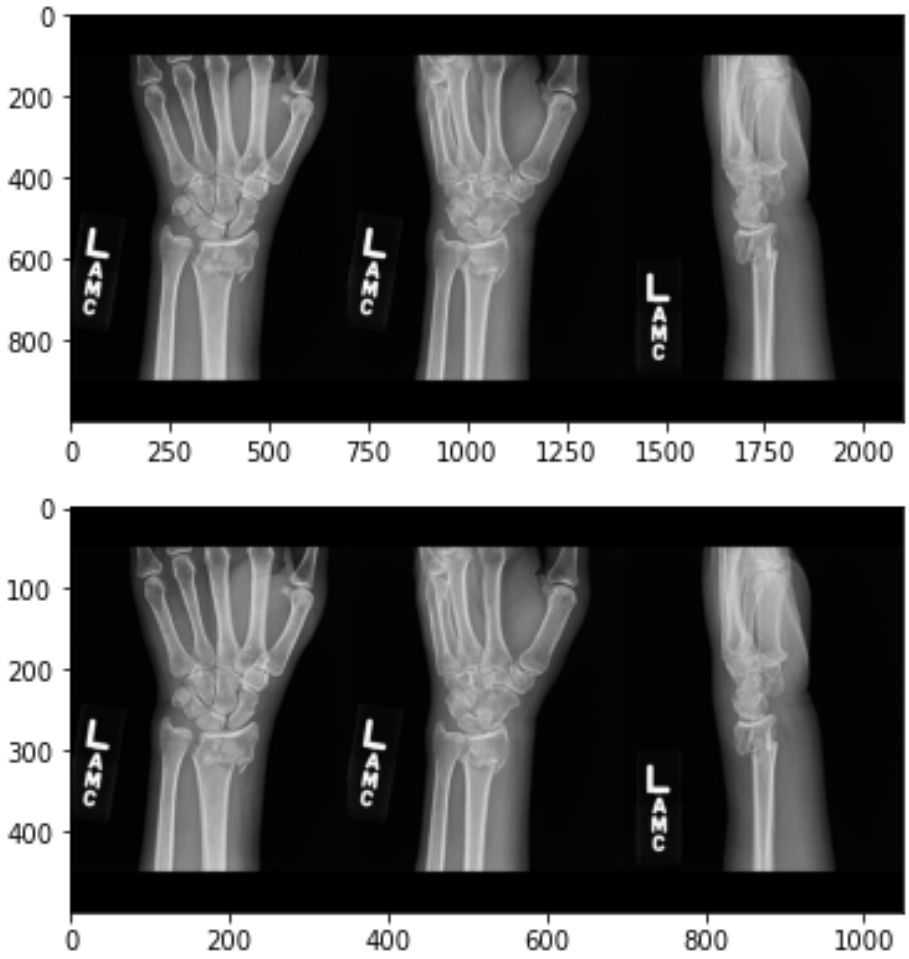


Figure 8. Full resolution and reduced resolution combined images.

Functional Requirements

The project was built in Python 3.8 using Jupyter notebooks and Python script files running in the Spyder environment.

Convolutional Neural Network: A Jupyter notebook of the CNN takes images from 270 of the 300 classified cases in CSV format and trains and validates a model. A set of 30 classified cases was set aside for testing the model. The dataset is unbalanced. There are only 12 instances of type III fractures in the set of 300 cases. One of the type III fractures was placed in the test set. The remaining 11 type III fractures were augmented by shifting the crop window slightly on the original subjects. The model was trained and validated on a set of 281 images with 11 augmented type III fractures. The model parameters and weights were saved and used in the classifier.

Classifier: The classifier imports three views of a distal radius fracture in DICOM format and classifies the fracture according to the system developed at the University of New Mexico.

Source Code Listing

Jupyter notebook for reading DICOM images, cropping and converting to CSV to make the training/validation set and test set

Import packages and define DICOM reading function

```
import numpy as np
import pandas as pd
import PySimpleGUI as sg
import pydicom
from pydicom.pixel_data_handlers.util import apply_voi_lut

import matplotlib.pyplot as plt
%matplotlib inline


def read_xray(path, voi_lut = True, fix_monochrome = True):
    dicom = pydicom.read_file(path)

    # VOI LUT (if available by DICOM device) is used to transform raw DICOM
data to "human-friendly" view
    if voi_lut:
        data = apply_voi_lut(dicom.pixel_array, dicom)
    else:
        data = dicom.pixel_array

    # depending on this value, X-ray may look inverted - fix that:
    if fix_monochrome and dicom.PhotometricInterpretation == "MONOCHROME1":
        data = np.amax(data) - data

    data = data - np.min(data)
    data = data / np.max(data)
    data = (data * 255).astype(np.uint8)

    return data
```

Get filename for saving the files

```
layout = [  [sg.Text("What is the subject number?")],
            [sg.Input()],
            [sg.Button('Ok')] ]

window = sg.Window('Subject Number', layout)


event, values = window.read()

SubNum = values[0]

filenameOutput1 = "Subject"+SubNum+"View1.csv"
```

```
filenameOutput2 = "Subject"+SubNum+"View2.csv"
filenameOutput3 = "Subject"+SubNum+"View3.csv"

window.close()

print(SubNum)
```

Ask whether the images are for a left hand or a right hand

```
layout = [  [sg.Text("Are the images for a left hand or a right hand?
(L/R)")],
            [sg.Input()],
            [sg.Button('Ok')] ]

window = sg.Window('Left or Right', layout)


event, values = window.read()

Hand01 = values[0]

window.close()
```

Get filenames for DICOM images

```
filename01 = sg.popup_get_file('Enter the file for the PA View')
filename02 = sg.popup_get_file('Enter the file for the Oblique View')
filename03 = sg.popup_get_file('Enter the file for the Lateral View')
```

Load and look at the images and determine where to crop the images

PA View

```
img1 = read_xray(filename01)
plt.figure(figsize = (12,12))
plt.imshow(img1, 'gray')
plt.title("PA View")
plt.savefig('C:/Classifier/PAView.png')

# Determine crop height

layoutPAH =  [   [sg.Text("What is the crop height? 0 start from top, 0.5
centered, 1 start at bottom")],
                 [sg.Input()],
                 [sg.Button('Ok')],
                 [sg.Image('C:/Classifier/PAView.png')] ]

window = sg.Window("Crop Height", layoutPAH)

event, values = window.read()
```

```
CHPA = values[0]
CHPA = float(CHPA)

window.close()

# Determine crop width

layoutPAW =  [   [sg.Text("What is the crop width? 0 start from left, 0.5
centered, 1 start at right")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image('C:/Classifier/PAView.png')] ]

window = sg.Window("Crop Width", layoutPAW)

event, values = window.read()

CWPA = values[0]
CWPA = float(CWPA)

window.close()

# plt.figure(figsize = (12,12))
# plt.show

# Crop images and reverse right hands so that the radius is always on the
right of the image

x1 = img1.shape
PAH0 = np.round(CHPA*(x1[0]-1000)).astype(int)
PAW0 = np.round(CWPA*(x1[1]-700)).astype(int)

img1cr = img1[PAH0:PAH0+1000,
              PAW0:PAW0+700]

if Hand01=="R" or Hand01=="r":
    img1crpd = pd.DataFrame(img1cr)
    img1cr = pd.DataFrame(img1crpd.iloc[:,699])
    for i in range(699):
        img1cr = img1cr.join(pd.DataFrame(img1crpd.iloc[:,698-i]))
```

Look at the cropped image to make sure the fracture is well-placed

```
plt.figure(figsize = (12,12))
plt.imshow(img1cr, 'gray')
```

Oblique View

```
# Determine crop height

img2 = read_xray(filename02)
plt.figure(figsize = (12,12))
```

```
plt.imshow(img2, 'gray')
plt.title("Oblique Veiw")
plt.savefig('C:/Classifier/ObView.png')


layoutObH =  [   [sg.Text("What is the crop height? 0 start from top, 0.5
centered, 1 start at bottom")],
                 [sg.Input()],
                 [sg.Button('Ok')],
                 [sg.Image( 'C:/Classifier/ObView.png')] ]

window = sg.Window("Crop Height", layoutObH)

event, values = window.read()

CHOb = values[0]
CHOb = float(CHOb)

window.close()

# Determine crop width

layoutObW =  [   [sg.Text("What is the crop width? 0 start from left, 0.5
centered, 1 start at right")],
                 [sg.Input()],
                 [sg.Button('Ok')],
                 [sg.Image( 'C:/Classifier/ObView.png')] ]

window = sg.Window("Crop Width", layoutObW)

event, values = window.read()

CWOb = values[0]
CWOb = float(CWOb)

window.close()

x2 = img2.shape
ObH0 = np.round(CHOb*(x2[0]-1000)).astype(int)
ObW0 = np.round(CWOb*(x2[1]-700)).astype(int)

img2cr = img2[ObH0:ObH0+1000,
              ObW0:ObW0+700]

if Hand01=="R" or Hand01=="r":
    img2crpd = pd.DataFrame(img2cr)
    img2cr = pd.DataFrame(img2crpd.iloc[:,699])
    for i in range(699):
        img2cr = img2cr.join(pd.DataFrame(img2crpd.iloc[:,698-i]))
```

Look at the cropped image to make sure the fracture is well-placed

```
plt.figure(figsize = (12,12))
plt.imshow(img2cr, 'gray')
```

Lateral View

```
img3 = read_xray(filename03)
plt.figure(figsize = (12,12))
plt.imshow(img3, 'gray')
plt.title("Lateral View")
plt.savefig('C:/Classifier/LView.png')

# Determine crop height

layoutLH =  [   [sg.Text("What is the crop height? 0 start from top, 0.5
centered, 1 start at bottom")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image( 'C:/Classifier/LView.png')] ]

window = sg.Window("Crop Height", layoutLH)

event, values = window.read()

CHL = values[0]
CHL = float(CHL)

window.close()

# Determine crop width

layoutLW =  [   [sg.Text("What is the crop width? 0 start from left, 0.5
centered, 1 start at right")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image( 'C:/Classifier/LView.png')] ]

window = sg.Window("Crop Width", layoutLW)

event, values = window.read()

CWL = values[0]
CWL = float(CWL)

window.close()


x3 = img3.shape
LH0 = np.round(CHL*(x3[0]-1000)).astype(int)
LW0 = np.round(CWL*(x3[1]-700)).astype(int)

img3cr = img3[LH0:LH0+1000,
              LW0:LW0+700]
```

```
if Hand01=="R" or Hand01=="r":
    img3crpd = pd.DataFrame(img3cr)
    img3cr = pd.DataFrame(img3crpd.iloc[:,699])
    for i in range(699):
        img3cr = img3cr.join(pd.DataFrame(img3crpd.iloc[:,698-i]))
```

Look at the cropped image to make sure the fracture is well-placed

```
plt.figure(figsize = (12,12))
plt.imshow(img3cr, 'gray')
```

Save the cropped images

```
np.savetxt(filenameOutput1, img1cr, delimiter=",")
np.savetxt(filenameOutput2, img2cr, delimiter=",")
np.savetxt(filenameOutput3, img3cr, delimiter=",")
```

Jupyter notebook for data preparation of the training/validation set for the CNN

Import packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Load data: Load subjects 1 through 201 and 232 through 300 for training and validation. Cases 202 through 231 will be the test set.

Load the first case.

```
view1 = np.asarray(pd.read_csv('Subject001View1.csv', header = None))
view2 = np.asarray(pd.read_csv('Subject001View2.csv', header = None))
view3 = np.asarray(pd.read_csv('Subject001View3.csv', header = None))
```

Append cases 2 through 9

```
s1 = 'Subject00'
s21 = 'View1.csv'
s22 = 'View2.csv'
s23 = 'View3.csv'
for i in range(8):
    # view 1
    filename1 = (s1+str(i+2)+s21)
    temp11 = pd.read_csv(filename1, header = None)
    temp12 = np.asarray(temp11)
    view1 = np.vstack((view1, temp12))
    # view 2
    filename2 = (s1+str(i+2)+s22)
    temp21 = pd.read_csv(filename2, header = None)
```

```
    temp22 = np.asarray(temp21)
    view2 = np.vstack((view2, temp22))
    # view 3
    filename3 = (s1+str(i+2)+s23)
    temp31 = pd.read_csv(filename3, header = None)
    temp32 = np.asarray(temp31)
    view3 = np.vstack((view3, temp32))
```

Append cases 10 through 99

```
s1 = 'Subject0'
s21 = 'View1.csv'
s22 = 'View2.csv'
s23 = 'View3.csv'
for i in range(90):
    # view 1
    filename1 = (s1+str(i+10)+s21)
    temp11 = pd.read_csv(filename1, header = None)
    temp12 = np.asarray(temp11)
    view1 = np.vstack((view1, temp12))
    #view 2
    filename2 = (s1+str(i+10)+s22)
    temp21 = pd.read_csv(filename2, header = None)
    temp22 = np.asarray(temp21)
    view2 = np.vstack((view2, temp22))
    # view 3
    filename3 = (s1+str(i+10)+s23)
    temp31 = pd.read_csv(filename3, header = None)
    temp32 = np.asarray(temp31)
    view3 = np.vstack((view3, temp32))
```

Append cases 100 through 201

```
s1 = 'Subject'
s21 = 'View1.csv'
s22 = 'View2.csv'
s23 = 'View3.csv'
for i in range(102):
    # view 1
    filename1 = (s1+str(i+100)+s21)
    temp11 = pd.read_csv(filename1, header = None)
    temp12 = np.asarray(temp11)
    view1 = np.vstack((view1, temp12))
    #view 2
    filename2 = (s1+str(i+100)+s22)
    temp21 = pd.read_csv(filename2, header = None)
    temp22 = np.asarray(temp21)
    view2 = np.vstack((view2, temp22))
    # view 3
    # print(i)
    filename3 = (s1+str(i+100)+s23)
    temp31 = pd.read_csv(filename3, header = None)
```

```
    temp32 = np.asarray(temp31)
    view3 = np.vstack((view3, temp32))
```

Append cases 232 through 311

```
s1 = 'Subject'
s21 = 'View1.csv'
s22 = 'View2.csv'
s23 = 'View3.csv'
for i in range(80):
    # view 1
    filename1 = (s1+str(i+232)+s21)
    temp11 = pd.read_csv(filename1, header = None)
    temp12 = np.asarray(temp11)
    view1 = np.vstack((view1, temp12))
    #view 2
    filename2 = (s1+str(i+232)+s22)
    temp21 = pd.read_csv(filename2, header = None)
    temp22 = np.asarray(temp21)
    view2 = np.vstack((view2, temp22))
    # view 3
    # print(i)
    filename3 = (s1+str(i+232)+s23)
    temp31 = pd.read_csv(filename3, header = None)
    temp32 = np.asarray(temp31)
    view3 = np.vstack((view3, temp32))
```

Make a combined with all three views for each case together

```
comboViews = np.append(view1, view2, axis = 1)
comboViews = np.append(comboViews, view3, axis = 1)
comboViews.shape
```

Out[12]:
```
(281000, 2100)
```

Save the array to a CSV file

```
np.savetxt('ComboViewAugTrainVal.csv', comboViews, delimiter=",")
```

Reshape the array for analysis

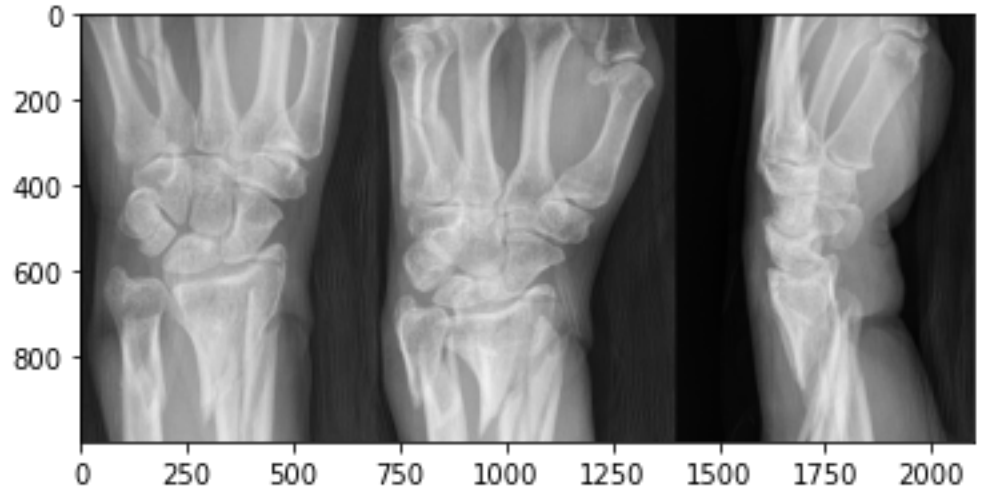
```
comboViewsNew = np.reshape(comboViews,(281,1000,2100,1))
```

View one image to make sure everything appended correctly

```
img = comboViewsNew[280,:,:]
plt.imshow(img, cmap='gray')
plt.show
```

Out[16]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Reduce the resolution of the images by ¼

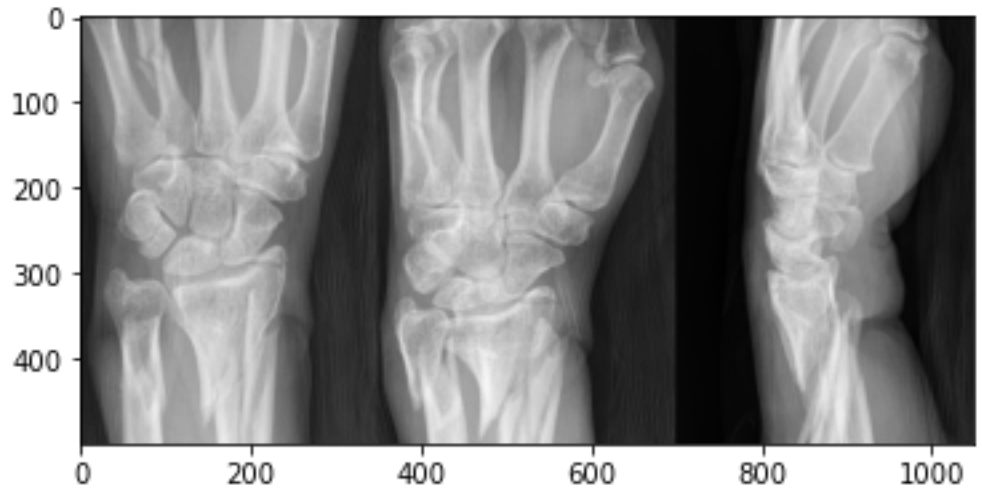
```
ComboReduce = np.zeros((140500, 1050))
ComboReduce.shape

for k in range(140500):
    for i in range(1050):
        ComboReduce[k,i] = (comboViews[2*k,2*i,]+comboViews[2*k+1,2*i]+
                                    comboViews[2*k,
2*i+1]+comboViews[2*k+1,2*i+1])/4
```

Look at an image to see if it reduced correctly

```
ComboReduceNew = np.reshape(ComboReduce,(281,500,1050,1))
img1 = ComboReduceNew[280,:,:]
plt.imshow(img1, cmap='gray')
plt.show
```

Out[21]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Save the reduced array

```
np.savetxt('ComboReduceAugTrainVal.csv', ComboReduce, delimiter=",")
```

Jupyter notebook for data preparation of the test set for the CNN

Import packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Load data: Load 202 through 231.

Load the first case.

```
view1 = np.asarray(pd.read_csv('Subject202View1.csv', header = None))
view2 = np.asarray(pd.read_csv('Subject202View2.csv', header = None))
view3 = np.asarray(pd.read_csv('Subject202View3.csv', header = None))
```

Append cases 203 through 231

```
s1 = 'Subject'
s21 = 'View1.csv'
s22 = 'View2.csv'
s23 = 'View3.csv'
for i in range(29):
    # view 1
    filename1 = (s1+str(i+203)+s21)
    temp11 = pd.read_csv(filename1, header = None)
    temp12 = np.asarray(temp11)
    view1 = np.vstack((view1, temp12))
    #view 2
    filename2 = (s1+str(i+203)+s22)
```

```
    temp21 = pd.read_csv(filename2, header = None)
    temp22 = np.asarray(temp21)
    view2 = np.vstack((view2, temp22))
    # view 3
    # print(i)
    filename3 = (s1+str(i+203)+s23)
    temp31 = pd.read_csv(filename3, header = None)
    temp32 = np.asarray(temp31)
    view3 = np.vstack((view3, temp32))
```

Make the combined view

```
comboViews = np.append(view1, view2, axis = 1)
comboViews = np.append(comboViews, view3, axis = 1)
comboViews.shape
```

Out[7]:

```
(30000, 2100)
```

Save the array

```
np.savetxt('ComboViewAugTest.csv', comboViews, delimiter=",")
```

Reduce resolution of images by ¼

```
ComboReduce = np.zeros((15000, 1050))

for k in range(15000):
    for i in range(1050):
        ComboReduce[k,i] = (comboViews[2*k,2*i,]+comboViews[2*k+1,2*i]+
                                    comboViews[2*k,
2*i+1]+comboViews[2*k+1,2*i+1])/4
```

Save the reduced array

```
np.savetxt('ComboReduceAugTest.csv', ComboReduce, delimiter=",")
```

Jupyter notebook for the Convolutional Neural Network

Import packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPool2D
from keras.models import model_from_json
from sklearn.model_selection import train_test_split
```

Load previously saved data

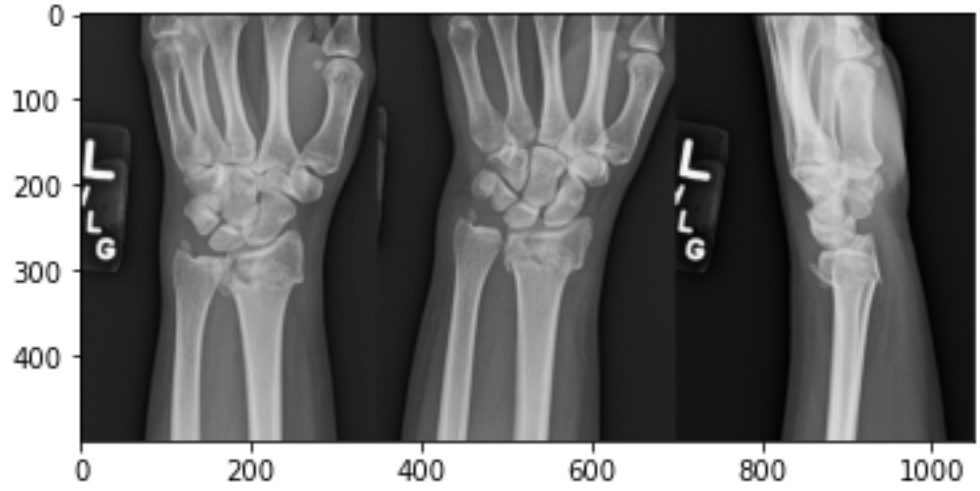
```
ComboReduce = np.asarray(pd.read_csv('ComboReduceAugTrainVal.csv', header =
None))
ComboReduce = np.reshape(ComboReduce, (281, 500, 1050, 1))
```

Look at the combination view to make sure it is as expected

```
img = ComboReduce[0,:,:,0]
plt.imshow(img, cmap='gray')
plt.show
```

Out[6]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Normalize the data from 0 to 255 to 0 to 1

```
ComboReduce = ComboReduce/255.0
```

Divide the data into a training set and validation set

```
ComboTrain, ComboVal, yTrain, yVal = train_test_split(ComboReduce,
DRUJLabels, test_size=0.20, stratify=DRUJLabels)

print("ComboTest shape = ", ComboVal.shape, "yTest shape = ", yVal.shape)
print("ComboTrain shape = ", ComboTrain.shape, "yTrain shape = ",
yTrain.shape)

ComboTest shape =  (57, 500, 1050, 1) yTest shape =  (57, 3)
ComboTrain shape =  (224, 500, 1050, 1) yTrain shape =  (224, 3)
```

Create the model and add layers

```
model = Sequential()
model.add(Conv2D(128, kernel_size=5, activation='relu',
input_shape=(500,1050,1)))
model.add(Conv2D(64, kernel_size=5, activation='relu'))
model.add(Conv2D(32, kernel_size=5, activation='relu'))
model.add(Conv2D(16, kernel_size=5, activation='relu'))

# model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))

model.add(Dropout(0.1))

model.add(Flatten())

# model.add(Dense(27, activation='relu'))

model.add(Dense(3, activation='softmax'))
```

Compile the model. Use accuracy to measure model performance

```
opt = tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(optimizer=opt, loss=tf.keras.losses.categorical_crossentropy,
metrics=['accuracy'])

model.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 496, 1046, 128)    3328
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 492, 1042, 64)     204864
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 488, 1038, 32)     51232
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 484, 1034, 16)     12816
_________________________________________________________________
```

```
dropout (Dropout)            (None, 484, 1034, 16)     0
_________________________________________________________________
flatten (Flatten)            (None, 8007296)           0
_________________________________________________________________
dense (Dense)                (None, 3)                 24021891
=================================================================
Total params: 24,294,131
Trainable params: 24,294,131
Non-trainable params: 0
_________________________________________________________________
```

Train the model with 30 epochs

```
checkpoint = tf.keras.callbacks.ModelCheckpoint("model20220202a.h5",
monitor='val_accuracy', verbose=1, save_best_only=True,
save_weights_only=True, mode='auto')
early = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0,
patience=20, verbose=1, mode='auto')

history = model.fit(ComboTrain, yTrain, epochs=30, validation_data=(ComboVal,
yVal), callbacks=[checkpoint])
```

Look at the results

```
predict_x=model.predict(ComboVal)
classes_x=np.argmax(predict_x,axis=1)
print(classes_x)
print(predict_x)

DRUJLabelsVal = []

for i in range(len(yVal)):
    if yVal[i,0]==1:
        temp = 0
    if yVal[i,1]==1:
        temp = 1
    if yVal[i,2]==1:
        temp = 2
    DRUJLabelsVal = np.append(DRUJLabelsVal,temp)

print(DRUJLabelsVal)

DRUJLabelsTrain = []

for i in range(len(yTrain)):
    if yTrain[i,0]==1:
        temp = 0
    if yTrain[i,1]==1:
        temp = 1
    if yTrain[i,2]==1:
```

```
        temp = 2
    DRUJLabelsTrain = np.append(DRUJLabelsTrain,temp)

print(DRUJLabelsTrain)
```

Save the model parameters and weights

```
# serialize model to JSON
model_json = model.to_json()
with open("model20220202a.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model20220202a1.h5")
print("Saved model to disk")
```

Data visualizations for CNN Model – Plot Loss versus Epoch and Accuracy versus Epoch

```
history.history

loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,31)
plt.plot(epochs, loss_train, 'b', label='Training loss')
plt.plot(epochs, loss_val, 'g', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig('Loss20220202a.png')

acc_train = history.history['accuracy']
acc_val = history.history['val_accuracy']
epochs = range(1,31)
plt.plot(epochs, acc_train, 'g', label='Training accuracy')
plt.plot(epochs, acc_val, 'b', label='validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('Accuracy202200202a.png')
```

Save the history to a csv file

```
lossacc = np.stack((loss_train, loss_val, acc_train, acc_val), axis = 1)
np.savetxt('LossAcc202200202a.csv', lossacc, delimiter=",")
```

Confusion matrices for Training Set and Validation Set

```
from sklearn.metrics import confusion_matrix

ConfMatrixModel = confusion_matrix(DRUJLabelsVal, classes_x)
```

```
ConfMatrixModeldf = pd.DataFrame(ConfMatrixModel,
                     index = ['I','II','III'],
                     columns = ['I','II','III'])
ConfMatrixModeldf['Actual Sum'] = ConfMatrixModeldf.sum(axis=1)
PredictedSum = ConfMatrixModeldf.sum(axis=0)
ConfMatrixModeldf.loc[len(ConfMatrixModeldf.index)] = [PredictedSum[0],
PredictedSum[1], PredictedSum[2], PredictedSum[3]]
ConfMatrixModeldf.index = ['I', 'II', 'III', 'Predicted Sum']
ConfMatrixModeldf

predict_xTrain=model.predict(ComboTrain)
classes_xTrain=np.argmax(predict_xTrain,axis=1)

ConfMatrixModel = confusion_matrix(DRUJLabelsTrain, classes_xTrain)
ConfMatrixModeldf = pd.DataFrame(ConfMatrixModel,
                     index = ['I','II','III'],
                     columns = ['I','II','III'])
ConfMatrixModeldf['Actual Sum'] = ConfMatrixModeldf.sum(axis=1)
PredictedSum = ConfMatrixModeldf.sum(axis=0)
ConfMatrixModeldf.loc[len(ConfMatrixModeldf.index)] = [PredictedSum[0],
PredictedSum[1], PredictedSum[2], PredictedSum[3]]
ConfMatrixModeldf.index = ['I', 'II', 'III', 'Predicted Sum']
ConfMatrixModeldf
```

Load model with best validation accuracy

```
# load json and create model
json_file = open('model202200202a.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights('model20220202a.h5')
print("Loaded model from disk")
```

Accuracy of saved model on Training Set

```
loaded_model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
score = loaded_model.evaluate(ComboTrain, yTrain, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

Accuracy of saved model on Validation Set

```
loaded_model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
score = loaded_model.evaluate(ComboVal, yVal, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

Accuracy of saved model on Test Set

```
ComboTest = np.asarray(pd.read_csv('ComboReduceAugTest.csv', header = None))
ComboTest = ComboTest/255
yTest = np.asarray(pd.read_csv('DRUJLabelsOneHotTest.csv', header = None))

DRUJLabelsTest = []

for i in range(len(yTest)):
    if yTest[i,0]==1:
        temp = 0
    if yTest[i,1]==1:
        temp = 1
    if yTest[i,2]==1:
        temp = 2
    DRUJLabelsTest = np.append(DRUJLabelsVal,temp)


loaded_model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
score = loaded_model.evaluate(ComboTest, yTest, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

Confusion matrices for Training Set, Validation Set, and Test Set

```
predict_LxVal=loaded_model.predict(ComboVal)
classes_LxVal=np.argmax(predict_LxVal,axis=1)

ConfMatrixModel = confusion_matrix(DRUJLabelsVal, classes_LxVal)
ConfMatrixModeldf = pd.DataFrame(ConfMatrixModel,
                    index = ['I','II','III'],
                    columns = ['I','II','III'])
ConfMatrixModeldf['Actual Sum'] = ConfMatrixModeldf.sum(axis=1)
PredictedSum = ConfMatrixModeldf.sum(axis=0)
ConfMatrixModeldf.loc[len(ConfMatrixModeldf.index)] = [PredictedSum[0],
PredictedSum[1], PredictedSum[2], PredictedSum[3]]
ConfMatrixModeldf.index = ['I', 'II', 'III', 'Predicted Sum']
ConfMatrixModeldf

predict_LxTrain=loaded_model.predict(ComboTrain)
classes_LxTrain=np.argmax(predict_LxTrain,axis=1)

ConfMatrixModel = confusion_matrix(DRUJLabelsTrain, classes_LxTrain)
ConfMatrixModeldf = pd.DataFrame(ConfMatrixModel,
                    index = ['I','II','III'],
                    columns = ['I','II','III'])
ConfMatrixModeldf['Actual Sum'] = ConfMatrixModeldf.sum(axis=1)
PredictedSum = ConfMatrixModeldf.sum(axis=0)
ConfMatrixModeldf.loc[len(ConfMatrixModeldf.index)] = [PredictedSum[0],
PredictedSum[1], PredictedSum[2], PredictedSum[3]]
ConfMatrixModeldf.index = ['I', 'II', 'III', 'Predicted Sum']
ConfMatrixModeldf
```

```
predict_LxTest=loaded_model.predict(ComboTest)
classes_LxTest=np.argmax(predict_LxTest,axis=1)

ConfMatrixModel = confusion_matrix(DRUJLabelsTest, classes_LxTest)
ConfMatrixModeldf = pd.DataFrame(ConfMatrixModel,
                     index = ['I','II','III'],
                     columns = ['I','II','III'])
ConfMatrixModeldf['Actual Sum'] = ConfMatrixModeldf.sum(axis=1)
PredictedSum = ConfMatrixModeldf.sum(axis=0)
ConfMatrixModeldf.loc[len(ConfMatrixModeldf.index)] = [PredictedSum[0],
PredictedSum[1], PredictedSum[2], PredictedSum[3]]
ConfMatrixModeldf.index = ['I', 'II', 'III', 'Predicted Sum']
ConfMatrixModeldf
```

Python Code for Classifier. Runs in the Spyder environment.

```
                # -*- coding: utf-8 -*-
"""
Created on Tue Nov 30 09:40:03 2021

@author: AnLWells

DRUJ Classifier
"""

# Load required packages

import pandas as pd
import PySimpleGUI as sg
import matplotlib.pyplot as plt
from keras.models import model_from_json
import numpy as np

# Define a function for reading DICOM Images

import pydicom
from pydicom.pixel_data_handlers.util import apply_voi_lut


def read_xray(path, voi_lut = True, fix_monochrome = True):
    dicom = pydicom.read_file(path)

    # VOI LUT (if available by DICOM device) is used to transform raw DICOM
data to "human-friendly" view
    if voi_lut:
        data = apply_voi_lut(dicom.pixel_array, dicom)
    else:
        data = dicom.pixel_array

    # depending on this value, X-ray may look inverted - fix that:
    if fix_monochrome and dicom.PhotometricInterpretation == "MONOCHROME1":
        data = np.amax(data) - data

    data = data - np.min(data)
    data = data / np.max(data)
    data = (data * 255).astype(np.uint8)

    return data

# Get path prefix for reading and saving files

layout = [  [sg.Text("What is the path where you will store files? For
example: C:/Classifier/")],
            [sg.Input()],
            [sg.Button('Ok')] ]
```

```
window = sg.Window('Path Prefix', layout)


event, values = window.read()

filenamePrefix = values[0]
filenameCombView = filenamePrefix+"myFigure.png"

window.close()

# Ask whether the images are for a left hand or a right hand

layout = [  [sg.Text("Are the images for a left hand or a right hand?
(L/R)")],
            [sg.Input()],
            [sg.Button('Ok')] ]

window = sg.Window('Left or Right', layout)


event, values = window.read()

Hand01 = values[0]

window.close()

# Filenames for DICOM images

filename01 = sg.popup_get_file('Enter the file for the PA View')
filename02 = sg.popup_get_file('Enter the file for the Oblique View')
filename03 = sg.popup_get_file('Enter the file for the Lateral View')

# Load and look at the images
# Determine where to crop the images

img1 = read_xray(filename01)
plt.figure(figsize = (12,12))
plt.imshow(img1, 'gray')
plt.title("PA View")
plt.savefig(filenamePrefix + 'PAView.png')

# Determine crop height

layoutPAH =  [   [sg.Text("What is the crop height? 0 start from top, 0.5
centered, 1 start at bottom")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image(filenamePrefix+'PAView.png')] ]

window = sg.Window("Crop Height", layoutPAH)

event, values = window.read()
```

```
CHPA = values[0]
CHPA = float(CHPA)

window.close()

# Determine crop width

layoutPAW =  [   [sg.Text("What is the crop width? 0 start from left, 0.5
centered, 1 start at right")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image(filenamePrefix+'PAView.png')] ]

window = sg.Window("Crop Width", layoutPAW)

event, values = window.read()

CWPA = values[0]
CWPA = float(CWPA)

window.close()


img2 = read_xray(filename02)
plt.figure(figsize = (12,12))
plt.imshow(img2, 'gray')
plt.title("Oblique Veiw")
plt.savefig(filenamePrefix + 'ObView.png')

# Determine crop height

layoutObH =  [   [sg.Text("What is the crop height? 0 start from top, 0.5
centered, 1 start at bottom")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image(filenamePrefix+'ObView.png')] ]

window = sg.Window("Crop Height", layoutObH)

event, values = window.read()

CHOb = values[0]
CHOb = float(CHOb)

window.close()

# Determine crop width

layoutObW =  [   [sg.Text("What is the crop width? 0 start from left, 0.5
centered, 1 start at right")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image(filenamePrefix+'ObView.png')] ]
```

```
window = sg.Window("Crop Width", layoutObW)

event, values = window.read()

CWOb = values[0]
CWOb = float(CWOb)

window.close()


img3 = read_xray(filename03)
plt.figure(figsize = (12,12))
plt.imshow(img3, 'gray')
plt.title("Lateral View")
plt.savefig(filenamePrefix + 'LView.png')

# Determine crop height

layoutLH =  [   [sg.Text("What is the crop height? 0 start from top, 0.5
centered, 1 start at bottom")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image(filenamePrefix+'LView.png')] ]

window = sg.Window("Crop Height", layoutLH)

event, values = window.read()

CHL = values[0]
CHL = float(CHL)

window.close()

# Determine crop width

layoutLW =  [   [sg.Text("What is the crop width? 0 start from left, 0.5
centered, 1 start at right")],
                [sg.Input()],
                [sg.Button('Ok')],
                [sg.Image(filenamePrefix+'LView.png')] ]

window = sg.Window("Crop Width", layoutLW)

event, values = window.read()

CWL = values[0]
CWL = float(CWL)

window.close()


plt.figure(figsize = (12,12))
```

```
plt.show

# Crop images and reverse right hands so that the radius is always on the
right of the image

x1 = img1.shape
PAH0 = np.round(CHPA*(x1[0]-1000)).astype(int)
PAW0 = np.round(CWPA*(x1[1]-700)).astype(int)

img1cr = img1[PAH0:PAH0+1000,
              PAW0:PAW0+700]

if Hand01=="R" or Hand01=="r":
    img1crpd = pd.DataFrame(img1cr)
    img1cr = pd.DataFrame(img1crpd.iloc[:,699])
    for i in range(699):
        img1cr = img1cr.join(pd.DataFrame(img1crpd.iloc[:,698-i]))

x2 = img2.shape
ObH0 = np.round(CHOb*(x2[0]-1000)).astype(int)
ObW0 = np.round(CWOb*(x2[1]-700)).astype(int)

img2cr = img2[ObH0:ObH0+1000,
              ObW0:ObW0+700]

if Hand01=="R" or Hand01=="r":
    img2crpd = pd.DataFrame(img2cr)
    img2cr = pd.DataFrame(img2crpd.iloc[:,699])
    for i in range(699):
        img2cr = img2cr.join(pd.DataFrame(img2crpd.iloc[:,698-i]))


x3 = img3.shape
LH0 = np.round(CHL*(x3[0]-1000)).astype(int)
LW0 = np.round(CWL*(x3[1]-700)).astype(int)

img3cr = img3[LH0:LH0+1000,
              LW0:LW0+700]

if Hand01=="R" or Hand01=="r":
    img3crpd = pd.DataFrame(img3cr)
    img3cr = pd.DataFrame(img3crpd.iloc[:,699])
    for i in range(699):
        img3cr = img3cr.join(pd.DataFrame(img3crpd.iloc[:,698-i]))


# Make combined view of images

ComboView = np.append(img1cr, img2cr, axis=1)
ComboView = np.append(ComboView, img3cr, axis=1)

# Reduce resolution of combined view
```

```
ComboReduce = np.zeros((500, 1050))
ComboViewN = ComboView/255

for k in range(500):
    for i in range(1050):
        ComboReduce[k,i] = round((ComboViewN[2*k,2*i]+ComboViewN[2*k+1,2*i]+
                                  ComboViewN[2*k,
                                    2*i+1]+ComboViewN[2*k+1,2*i+1])/4)

# View Combined view

plt.figure(figsize=(6, 3))
plt.imshow(ComboView, cmap='gray')
plt.title("Classified fracture, Combined View")
plt.savefig(filenameCombView)

# Reshape the input to match the input to the model

ComboReduce = np.reshape(ComboReduce,(1,500,1050,1))

# Load and run the model

# load json and create model

json_file = open(filenamePrefix+'model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

# load weights into new model

loaded_model.load_weights(filenamePrefix+"model.h5")
print(" ")
print("Loaded model from disk")

# evaluate loaded model on test data

loaded_model.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])

# Use the model to predict classification of fracture and produce the output
for the model

predict_x=loaded_model.predict(ComboReduce)
classes_x=np.argmax(predict_x,axis=1)

if classes_x==0:
    str01=" Type of fracture is I"
    str05=" A typical type I fracture is shown with the classified fracture
below"
    filename04 = filenamePrefix+'TypeI.png'
else:
    if classes_x==1:
```

```
        str01=" Type of fracture is II"
        str05=" A typical type II fracture is shown with the classified
fracture below"
        filename04 = filenamePrefix+'TypeII.png'
    else:
        str01=" Type of fracture is III"
        str05=" A typical type III fracture is shown with the classified
fracture below"
        filename04 = filenamePrefix+'TypeIII.png'

str02=' Type I:   ' + str(round(predict_x[0,0]*100,2))
str03=' Type II:  ' + str(round(predict_x[0,1]*100,2))
str04=' Type III: ' + str(round(predict_x[0,2]*100,2))

layout10 = [[sg.Listbox(values=[' Report for Classification of DRUJ
Fracture',
                                ' ', str01, ' ',
                                ' The softmax numbers for each type of
fracture are: ',
                                ' ', str02, str03, str04,
                                ' ', ' ', str05], size=(60,15),
font=('Times', 14))],
            [sg.Image(filename04)],
            [sg.Image(filenamePrefix+'myFigure.png')],
            [sg.Button('Ok')]]

window = sg.Window('Model Output',layout10)

event, values = window.read()

window.close()
```

Implementation Plan

The CNN will run on Google Colab using a High-RAM run time. The CNN will run on my Dell Precision 7750 which has 64 Gb of RAM. I would not expect the CNN to run on most laptops. It does not run on the desktop computers at University of New Mexico Health Sciences Center (UNM HSC).

The classifier will run on most computers that have Python installed. I am using the Spyder environment. When the files are downloaded from the Google drive, all of the files should be placed in the Classifier folder.

Working Project

All the executable files necessary for running the CNN and the Classifier are located on a Google drive linked here:
https://drive.google.com/drive/folders/1V7oJKbPplCAy19pMG7dlDXWhKDYokXhN?usp=sharing
I have set up the drive so that anyone with the link should have access to the folder. If the link does not work, copy and paste the address into your browser.

All the files that are small enough to be on GitHub are on a public repository linked here: https://github.com/IHGCU/CapstoneProject.

Files needed for running the CNN

CNNforDRUJCombinedViewReducedResolution.ipynp – Jupyter notebook

ComboReduceAugTrainVal.csv – CSV file with the reduced images for the 281 cases

DRUJLabelsAugTrainVal.csv – The one-hot encoded labels for the 281 cases

ComboReduceAugTest.csv – CSV file with the reduced images for the 30 cases

DRUJLabelsAugTest.csv – The one-hot encoded labels for the 30 cases

Files needed for running the classifier

DRUJClassifier.py – Python script for running the classifier

model.json – File containing the saved model parameters

model.h5 – File containing the saved model weights

TestCases – Folder with 5 test cases, each test case has three images

Classifier – Folder with three images of typical examples of Type I, Type II, and Type III fractures.

User Guide

The User Guide is a separate document and can be found on the Google drive and GitHub repository referenced above.

System Administration Guide

The System Administration Guide is a separate document and can be found on the Google drive and GitHub repository referenced above.

Testing

Component Testing

All the components run in Python Anaconda. The classifier runs in the Spyder environment of Anaconda. The code was reviewed by Jay T. Wells who has a doctorate in Computer Science.

Table 2. Component testing

Component	**Name of file**	**Dependencies**	**Result**
Data Preparation – Read DICOM images, crop and convert to CSV	CropImages.ipynb	Required packages: numpy, pandas, PySimpleGUI, pydicom, matplotlib	Jupyter notebook runs as expected, no issues
Data Preparation – Group images for training/validation set and testing set	DataPrepforDRUJ TrainVal.ipynb DataPrepforDRUJTest.ipynb	Required packages: numpy, pandas, matplotlib	Jupyter notebooks run as expected, no issues
Convolutional Neural Network Model builder	CNNforDRUJCombinedView ReducedResolution.ipynb	Required packages: numpy, pandas, matplotlib, random, tensorflow, keras, sklearn	Jupyter notebook runs as expected, no issues
Classifier	DRUJClassifier.py	Required packages: numpy, pandas, PySimpleGUI, matplotlib, keras,	Classifier runs as expected. Error checks need to be added to the user input requests so the classifier will not freeze.

Requirements Testing

Table 3. Requirements testing

Component: DICOM Reader and Converter		
Name of developer: A. Laurie Wells		
Checklist		
Type	Pass	Comments
Accurately crops and renders DICOM image to CSV file	Passed	Spot checked images were rendered correctly

Component: Gather images into a single file for feeding to CNN		
Name of developer: A. Laurie Wells		
Checklist		
Type	Pass	Comments
Loads reduced resolution images into a single file for reading by CNN	Passed	Spot checked images were rendered correctly

Component: CNN		
Name of developer: A. Laurie Wells		
Checklist		
Type	Pass	Comments
Accuracy of model	Passed	Model has an accuracy over 70%. A higher accuracy is desired, but a larger training set will be needed to improve accuracy.

Component: Classifier		
Name of developer: A. Laurie Wells		
Checklist		
Type	Pass	Comments
Consistency of classification	Passed	The classifier consistently classifies fractures. Consistency is better than human classifiers.

System Testing

This project is not part of a business enterprise. However, we can take a system view of the overall project.

This is a human research project. We have fulfilled all the requirements for human subject research under the University of New Mexico Health Sciences Center (UNM HSC) Human Research Protections Office (HRPO). The documentation of the approval is included with this submission. The research protocol is attached. The original approval, dated March 18, 2021, is 21-089 Mercer NS Approval Letter.pdf. The approval of the continuing review for this project, dated February 9, 2022, is 21-089 Mercer CR Approval Letter.pdf.

The project as approved requires the following data flow.

1. The Department of Orthopaedics administrator searched the electronic medical record for distal radius fractures using CPT Codes 25515, 25525, 25526, 25545, 25574, 25575, 25607, 25608, 25609, and 25652 and compiled a list of fractures treated at UNM HSC between the dates 01/01/2011 to 12/31/2020

2. A research assistant and a resident searched through the PACS system for subjects with three views of the injury before reduction. The images were downloaded as de-identified images to a UNM HSC computer.

3. The research assistant used a Python notebook to crop the images to 1000 x 700 pixels and save them as CSV files.

4. The cropped images were copied to PowerPoint files so that several orthopedic surgeons, residents, and medical students could classify the images.

5. A consensus classification was developed for each case from the classifications and this was used to train the model.

User Guide

Three people tested the user guide to run the classifier.

User 1: Novice. Chose the images in the wrong order for the first case. After one re-start she successfully classified the five test cases.

User 2: Computer professional – Doctorate in Computer Science. Successfully classified the five test cases.

User 3: Engineer. Successfully classified the five test cases.

System Administration Guide

The System Administration Guide can be found in both the Github repository and the Google drive linked below.

Github: https://github.com/IHGCU/CapstoneProject

Google drive: https://drive.google.com/drive/folders/1V7oJKbPplCAy19pMG7dlDXWhKDYokXhN?usp=sharing

Attachments

DSC 590 0500 Wells 21-089 Mercer CR Approval Letter.pdf – This is the approval by the IRB of the second year of research

DSC 590 0500 Wells 21-089 Mercer NS Approval Letter.pdf – This is original approval by the IRB of the research

DSC 590 0500 Wells Chart Review Protocol DRUJ Study Mercer.pdf – This is the research protocol for the overarching study

DSC 590 0500 Wells System Administration Guide for Distal Radius Fracture.pdf – This is the System Administration Guide

DSC 590 0500 Wells User Guide for Distal Radius Fracture Classifier.pdf – This is the user guide for the classifier

Conclusion

This project created a dataset of images of 300 classified distal radius fractures and a convolutional neural network model to classify the fractures from plain x-rays. Table 4 is a table of the parameters used in various iterations of the model. The parameter space was explored systematically, but there was not time to try every permutation. The model used for the published classifier is summarized in table 5.

Table 4. CNN Parameter space

CNN Parameter	Minimum Value	Maximum Value
Learning rate	0.0001	0.01
Layers	5	16
Dropout rate	0	0.3
Epochs	20	40
Kernel size	3	10
Nodes per layer (Conv2D layers)	512	16

Table 5. Model summary for classifier

Model: "sequential"		
Layer (type)	Output Shape	Parameter number
conv2d (Conv2D)	(None, 496, 1046, 128)	3328
conv2d_1 (Conv2D)	(None, 492, 1042, 64)	204846
conv2d_2 (Conv2D)	(None, 488, 1038, 32)	51232
conv2d_3 (Conv2D)	(None, 484, 1034, 16)	12816
dropout (Dropout)	(None, 484, 1034, 16)	0
flatten (Flatten)	(None, 8007296)	0
dense (Dense)	(None, 3)	24021891
Total parameters: 24,294,131 Trainable parameters: 24,294,131 Non-trainable parameters: 0 Epochs: 20		

The set of images used to train, validate, and test the model suffered from two major issues. First, the set of images was small. This led to overtraining that could not be completely resolved. Figure 9 shows training set and validation set accuracy for four iterations of the model. These iterations were trained for 30 epochs. It can be seen that the training set accuracy always outperformed the validation set accuracy. Figure 10 shows training set and validation set loss for four iterations of the model. The validation set loss begins to increase as the model experiences overtraining.

The second issue was that the set of images was unbalanced. Type I fractures made up 24% of the images. Type II fractures made up 72% of the images. Type III fractures made up only 4% of the images. Consequently, the model could not reliably classify type III fractures. For some iterations of the model, the training/validation set was augmented with shifted type III fractures. We doubled the number of type III fractures. This did not improve the model's ability to classify type III fractures, but it did lower the accuracy of the validation set even further.

Future work will focus on expanding the classified cases available for training the model. We will also try increasing the number of augmented type III fractures.

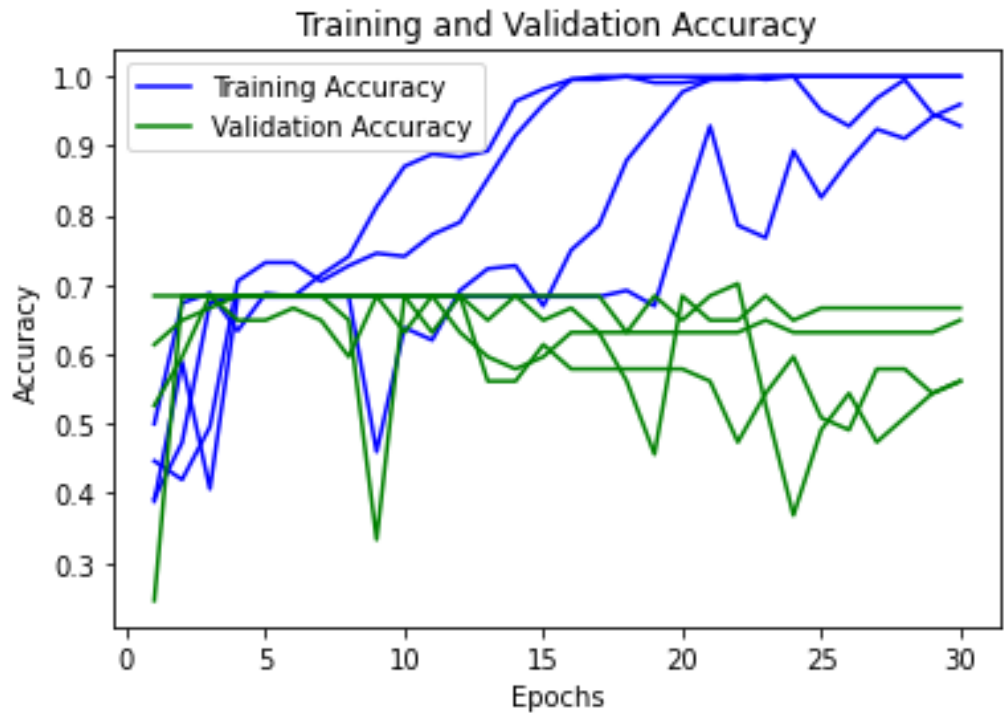


Figure 9. Training set and validation set accuracy.

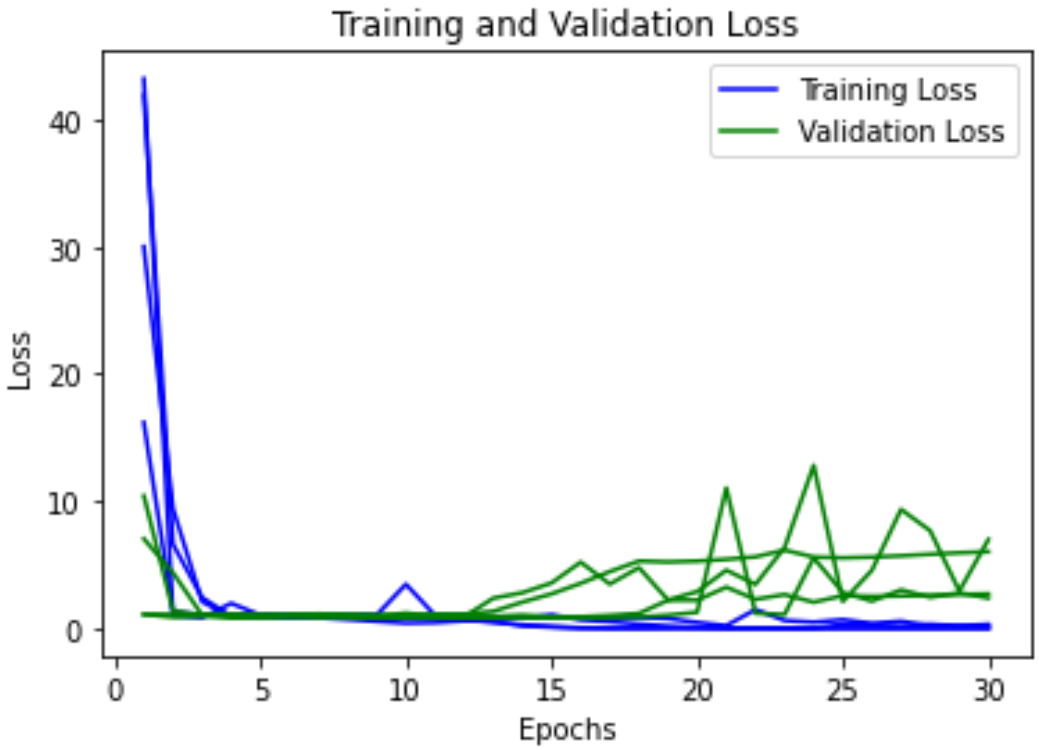


Figure 10. Training set and validation set loss.

References

Court-Brown, C. M., & Caesar, B. (2006). Epidemiology of adult fractures: A review. *Injury, 37*(8), 691–697. https://doi.org/10.1016/j.injury.2006.04.130

El-Baz, A., Gimel'farb, G., and Suzuki, K. (2017). Machine learning applications in medical image analysis. *Computational and Mathematical Methods in Medicine, 2017*. https://doi.org/10.1155/2017/2361061

Gan, K., Xu, D., Lin, Y., Shen, Y., Zhang, T., Hu, K., Zhou, K., Bi, M., Pan, L., Wu, W., & Liu, Y. (2019). Artificial intelligence detection of distal radius fractures: A comparison between the convolutional neural network and professional assessments. *Acta Orthopaedica*, *90*(4), 394–400. https://doi.org/10.1080/17453674.2019.1600125

Geissler, W. B., Fernandez, D. L., & Lamey, D. M. (1996). Distal radioulnar joint injuries associated with fractures of the distal radius. *Clinical orthopaedics and related research*, (327), 135–146. https://doi.org/10.1097/00003086-199606000-00018

Kandel, I. & Castelli, M. (2020). How Deeply to Fine-Tune a Convolutional Neural Network: A Case Study Using a Histopathology Dataset. *Applied Sciences. 10*. 3359. DOI 10.3390/app10103359.

Mercer, D., Heifner, J., Orbay, J., & Wells, A. L. (2021). Unpublished research protocol.

Packard, B., Yeager, K., Miller, R. A., Gavin, K., Sridhar, S., and Mlady, G. W. (2020). Inter- and intra-observer differences in proximal fifth metatarsal fractures. *Western Journal of Orthopaedics, 9*, 69-73. https://www.yumpu.com/en/document/read/63809274/western-journal-of-orthopaedics

Rundgren, J., Bojan, A., Mellstrand Navarro, C., & Enocson, A. (2020). Epidemiology, classification, treatment and mortality of distal radius fractures in adults: an observational

study of 23,394 fractures from the national Swedish fracture register. *BMC Musculoskelet Disord 21*(88). https://doi.org/10.1186/s12891-020-3097-8

Yadav, S.S. & Jadhav, S.M. (2019). Deep convolutional neural network based medical image classification for disease diagnosis. *J Big Data 6*, 113. https://doi.org/10.1186/s40537-019-0276-2